CDTV

# CDXL TOOLKIT DEVELOPER DOCUMENTATION

Version 1.0 – October 1991

---


**Overview**

**Warning – Read This**
- *Safety*
- *Performance*

**Commands**
- *XLMake*
- *XLPlay*
- *XLInfo*
- *XLJoin*
- *XLCopy*
- *XLTrim*
- *XLAudio*
- *XLClean*

**Creating Video**

**Adding Audio**

**Custom Programs**


---


From PANTARAY, Inc. Ukiah, CA
Software by Ken Yeast
Produced by Carl Sassenrath



© 1991 Commodore Electronics, Ltd.
All Rights Reserved.

# Overview

This package contains the first generation of CDXL tools to help you edit video/audio sequences and write programs to play the results. The distribution disks contain:

- Eight separate tool programs,
- A standard 'C' include file,
- An example source file,
- Example files in CDXL format.

Before using the tool programs, be certain to take the precautions stated in the next section.

The CDXL tools provided are:

*XLMake* creates or appends to a CDXL file. This tool accepts one or more IFF files (in the desired resolution and color mode), converts them to the standard CDXL format, and joins them together in sequence. Normally you would run this program as part of your video building script.

*XLPlay* simulates the playback of a CDXL file with approximate CDTV timing. XLPlay provides a simple means of viewing and hearing CDXL sequences on a standard Amiga computer as if they were on CDTV.

*XLInfo* displays detailed information about a CDXL file. The primary purpose of this tool is to provide size and timing information.

*XLJoin* combines multiple CDXL files into a single sequence. It can also insert a CDXL sequence into a second file at a particular point.

*XLCopy* copies frames from any point in a CDXL file and transfers them to a new file.

*XLTrim* removes frames from a CDXL file. Frames can be removed from any part of the file, or a file can be shortened overall by a specified amount.

*XLAudio* inserts an audio track into a CDXL file, ie., Sound on Film. The audio will become an integral part of the CDXL sequence and will be heard when you run XLPlay.

*XLClean* rebuilds the frame sequence numbers and linkage information within a CDXL file. This action is not normally required, but is sometimes useful if custom CDXL programs have been used.

Each of these tools will be described in detail shortly.

# Warning – Read This

CDXL sequences can be massive. For each minute of CDXL, nine megabytes of disk space are required. The CDXL Toolkit programs perform operations that are very read/write intensive. Should your Amiga crash (power failure, etc.) during an important operation, your AmigaDOS disk structures may become invalid and require reformatting.

**Special precautions must be taken.** It is well advised that you edit CDXL files on a separate disk drive, or at the very least, a separate partition of your drive. This has two advantages:

*Safety* If your system crashes during CDXL editing, you may lose some of your work in the disk/partition, but the rest of your system and code will not be affected. If you keep your application source code on the same system, this is a very important consideration.

*Performance* After a few days/weeks of CDXL editing, your disk/partition will become fragmented. Such fragmentation will introduce seek delays into your editing operations, which will cause the tools to slow down, and may result in poor timing and glitches in the XLPlay command.

So these are good arguments for taking the time to get set up properly. Believe us, it will be worth the effort! ***Partition your disk or use another disk for editing!!!***

On our single disk systems, what we normally do is: put the standard workbench, libraries, devices, etc. into a small (6 meg) partition; put our development tools and code into a second (40 meg) partition; and use the rest of the disk for CDXL editing.

# Commands

The CDXL Toolkit consists of a number of tools that operate as Shell (CLI) commands (they are not useful from the Workbench). Each of the tools require command arguments appropriate to its function. A brief summary of the arguments can be obtained by executing the command without any arguments.

Generally, each tool accepts both an input and an output file as arguments. The tool performs its function by reading the input file a frame at a time, making the required changes, and then writing to the output file. Normally, the input file is not modified (XLAudio has a special exception). This approach helps preserve your original files from accidental modification, but at the cost of extra disk space.

Note that the performance of the tools will vary depending on the speed of your Amiga processor and hard disk. The primary bottle neck, however, will usually be the AmigaDOS File System – large CDXL files may require significant time to process.

All of the tools deal with files that use a standard type of file format and header. Files that do not contain the proper header cannot be processed. This header is described later.

# XLMake

XLMake is the primary tool for creating CDXL sequences. The purpose of this tool is to accept standard IFF files that are already in the desired resolution, color mode and depth, translate them into a raw bitplane format, and join them together as a sequence of frames to be read with CDXL. Each frame contains header information about its size, resolution, colors, and audio.

This tool is normally used in a script file or ARexx program during the capture of individual video frames; for example, you might use it along with the Art Department Professional (from ASDG). Each time a new frame has been captured, scaled, and color adapted, XLMake is invoked to combine the frame to the CDXL file under construction. To help illustrate its use, a useful AREXX script has been provided on the distribution disks.

For this tool to run you must have "iffparse.library" installed on your system (in LIBS:). (We provide a copy on the distribution disk for those who are unfamiliar with this library).

*Summary*

```
XLMake [options] <IFFFile1> [<IFFFile2>...] <File>
```

The IFF files are translated and appended to the File. If the File does not exist, it will be created. If it does exist, new frames are appended to the end.

*Options*

**-a<n>** allocate space for audio track. The value specified is the number of bytes to use per frame. It is not necessary to allocate the audio during XLMake. You can always add the audio later with XLAudio (see XLAudio for more information). This feature is included in XLMake so you can view playback with accurate timing before your audio has been added.

**-d** forces AVM (Advanced Video Mode) and DCTV. If you are storing AVM pictures, this flag must be set.

**-f<n>** frame sequence number (not strictly required, but handy to know). Normally you would be assigning each frame number for every frame added; however, frame numbers are not required and XLClean can be used to renumber files later. When multiple IFF files are specified, the frame number is incremented automatically.

**-h** forces HAM mode (only needed for ancient Amiga programs that didn't set the CAMG correctly).

*Example*

```
XLMake -f10 -d intro.xl

XLMake -a1024 intro2.xl
```

# XLPlay

XLPlay is a very useful program that simulates the playback of a CDXL file with approximate CDTV timing. XLPlay provides a simple means of viewing and hearing CDXL sequences on a standard Amiga computer as if they were on CDTV. The purpose of this tool is to allow developers to view their CDXL files in order to make decisions and changes without remastering a CD (or creating an emulator version).

Playback timing is only approximate. The general timing characteristics are preserved (total play time), but the speed of AmigaDOS and the lack of CDXL on standard Amigas prevents precise timing from being achieved.

If an audio track is present, it will be played along with the video (mono only). Slight audio "drop–outs" will be heard due to the varying lengths of disk read times in the Amiga File System.

*Summary*

```
XLPlay [options] <File>
```

The specified File is played on the Amiga screen. The image will always be centered in the middle of the screen.

You can stop the play before it has finished by pressing the ESC key. To pause/unpause press the SPACE bar.

*Options*

**-h** force hi–res graphics mode (not normally required)

**-i** force interlace graphics mode (not normally required)

**-r<n>** repeat playback a number of times. This is helpful for short sequences that would be difficult to see.

**-q** quiet (mute the audio)

**-s[<n>]** show frame sequence numbers during playback. Handy if you need to know where to cut and edit. The optional numeric value indicates the "brightness" of the frame number display. It may range from 0 to 15, where 0 is black and 15 is white.

*Example*

```
XLPlay -r4 -s intro.xl
```

# XLInfo

XLInfo displays detailed information about a CDXL file. The primary purpose of this tool is to provide accurate timing and other information for application developers.

This tool will calculate and display the total play time, number of frames, frame sizes, frame resolutions, number of colors, color modes, audio track size, audio period estimate, sequencing information, and out of sequence frame numbers.

Normally only the first frame's header is examined, and an estimate is calculated from its size and the total file size. This is done to save time. To check the entire file, use the –w option.

*Summary*

```
XLInfo [options] <File>
```

CDXL file information and statistics will be displayed for the specified File.

*Options*

**–a[<n>]** estimate audio sizes. This will produce an table of what various audio track sizes would produce in terms of timing, playback period, total audio size, etc. If the optional value is provided, it will be interpreted as the number of bytes per frame for audio, and its relevant information will be shown.

**–f<n>** first frame to examine,

**–l<n>** last frame to examine

The above two options use the ranges to examine only a section of a XL file. Information and statistics will only pertain to this section.

**–w** examine whole file. Normally XLInfo just looks at the first frame and estimates the rest. If you have a mixture of frame sizes or wish to analyze the entire file, use this option.

*Example*

```
XLInfo -w intro.xl
```

# XLJoin

XLJoin combines multiple CDXL files into a single file. The purpose of this tool is to provide a means of splicing together separate video scenes to create a single CDXL video file.

XLJoin also allows the insertion of CDXL files anywhere within a CDXL file. This tool does not alter its input files in any way, thus preserving them for use in other video sequences.

*Summary*

```
XLJoin [options] <FileIn1> [<FileIn2>...] <FileOut>
```

Each of the input files will be appended (in the specified order) and written to the output file. Frames in the output file will have new sequence numbers.

*Options*

**-i<n>** insert into the first file at given frame. Normally XLJoin just combines a list of XL files into one XL file. This option allows you to "splice" the other files into the first input file (still producing the "out" file). This is useful when you wish to add an XL file into another "master" XL file.

*Example*

```
XLJoin intro.xl action.xl end.xl all.xl
```

# XLCopy

XLCopy copies frames from any point in a CDXL file and transfers them to a new file. The purpose of this tool is to provide a simple means of extracting useful video sequences without altering the original file. This approach is easier and safer than XLTrim in some situations.

*Summary*

```
XLCopy [options] <FileIn> <FileOut>
```

The specified portion of the input file will be copied to the output file. Frames in the output file will have new sequence numbers.

*Options*

**-f<n>** first frame to copy (inclusive). May be specified alone or in conjunction with the –l or –n options. When specified alone (without an end frame or length), frames through the end of the file are copied.

**-l<n>** last frame to copy (inclusive). May be specified alone or in conjunction with the –f or –n options. When specified alone (without a start frame or length), frames from the start of the file are copied. Using –n for a frame count will copy that many frames up to and including the last frame specified.

**-n<n>** number of frames to copy. This can be used with –f or -l but never with both.

*Example*

```
XLCopy -f10 -n100 intro.xl newintro.xl
```

# XLTrim

XLTrim supplies a simple set of editing functions for removing selected frames from CDXL files. It is useful for removing unwanted video or reducing the play time to some maximum value.

Options are provided to: trim video frames from the beginning, middle, or end; remove every Nth frame; and trim the entire file down to a specified number of frames or playback time.

This program also has the capability of removing the audio track from a CDXL file.

*Summary*

```
XLTrim [options] <FileIn> <FileOut>
```

The specified portion of the input file will be copied to the output file. Frames in the output file will have new sequence numbers.

*Options*

Options that specify frame numbers are inclusive.

**-a** remove the audio track. Actually removes the track and frees that space from each frame.

**-b<n>** remove from beginning to this frame

**-d<n>** delete every <n>th frame

**-e<n>** remove from the end back this number of frames. This option assumes that the file contains equal sized frames.

**-f<n>** first frame to start removing

**-l<n>** last frame to remove

**-n<n>** number of frames to remove

**-m<n>** maximum number of frames desired for entire file. Frames will be removed at evenly spaced intervals throughout the file in order to cut the file to this size. This option assumes that the file contains equal sized frames.

**-t<n>** maximum play time in seconds. Frames will be removed at evenly spaced intervals throughout the file in order to cut the file to this size. This option assumes that the file contains equal sized frames.

*Example*

```
XLTrim -f10 -n10 intro.xl short.xl
      (removes frames 10-19)
XLTrim -m60 intro.xl intro60.xl
      (removes frames to make result 60 seconds)
XLTrim -e5 intro.xl introcut.xl
      (removes the last 5 frames)
```

# XLAudio

XLAudio adds an audio sound track to an existing CDXL file. This operation is normally done as the last step in creating a CDXL audio/video sequence. The audio must already be adjusted to the correct sampling rate before laying down the track (see discussion in later section).

The tool will allocate the audio track for each CDXL frame if necessary, or overwrite the track if one already exists.

Only mono sound tracks are supported by this tool.

For this tool to run you must have "iffparse.library" installed on your system (in LIBS:). (We provide a copy on the distribution disk for those who are unfamiliar with this library).

*Summary*

```
XLAudio [options] <IFFSoundFile> <FileIn> [<FileOut>]
```

The IFF File is a sound file in either normal or compressed formats. FileIn holds the video sequence and may or may not already have an audio track. The output file contains both audio and video.

All frames in the input file must be of the same size.

*Options*

**-a<n>** allocate space for audio track. The value specified is the number of bytes to use per frame. See the discussion in a later section to determine the correct size. Also, the XLInfo -a option will print a table of possible values to consider using.

**-i** insert audio into existing track. This action operates on the input file and no output file is required. The frames in the input file must already have space allocated for the audio track (from XLMake or a previous XLAudio). This option is useful when the XL file is extremely large.

**-e** ignore audio size errors when inserting. When inserting audio with the -i option, XLAudio assumes that the audio track will remain the same size. When it does not, XLAudio complains unless this option is set.

**-p** preserve audio compression. The IFF sound file is copied with its compression intact.

*Example*

```
XLAudio -a1288 audio.iff video.xl final.xl
```

# XLClean

XLClean rebuilds the frame sequence numbers and linkage information within a CDXL file. This action is not normally required, but is sometimes useful if custom CDXL programs have been used.

*Summary* `XLClean <FileIn> <FileOut>`

The input file is cleaned up and written to the output file.

*Example* `XLClean grab.xl video.xl`

# Creating Video

To a large extent, the techniques you use for creating video will depend on the equipment you have at your disposal. A wide range of possibilities exist depending on your budget, ingenuity, and patience.

Of course the simplest solution is to directly generate the video graphics on your Amiga by using your favorite 2D painting or 3D modeling/rendering program. The only requirement is that the program be capable of producing displayable IFF files as output. Each frame would be stored as a separate file (on hard disk or on ram disk), and input to XLMake as needed.

Another approach might be to use a video frame digitizer that can be operated either manually or automatically (from AREXX or ADPRO). The images from this digitizer could be processed and altered with ADPRO then saved as an IFF file and again entered as input to XLMake. An simple AREXX script to do this might be:

```
/* CDXL ADPRO program for PPS Framegrabber: */
OPTIONS RESULTS
ADDRESS "ADPro"
LFORMAT "FRAMEGRABBER"
SFORMAT "IFF"
GETNUMBER '"How many frames?"'
if RC ~= 0 then exit
FRAMES = ADPRO_RESULT
DO I = 1 to FRAMES
      ADDRESS "ADPro"
      OKAY2 '"Ready to grab?"'
      if RC = 0 then exit
      LOAD "XXX"
      SCREEN_TYPE 0
      ABS_SCALE 160 100
      EXECUTE
      SAVE "RAM:TempFrame" "SCREEN"
      ADPRO_DISPLAY
      PAUSE 100
      ADPRO_UNDISPLAY

      ADDRESS Command
      XLMAKE "RAM:TempFrame" FILENAME
      END
```

# Adding Audio

Suppose want to make a short video clip....

60 sec Total Run Time

14 K/sec audio quality (approx)

12168 bytes/frame (from XLInfo)

First determine what you know for sure:

*Total Size:*

**60 sec * 150 K/sec * 1024 bytes/K = 9216000 bytes**

*Total Audio Size:*

**60 sec * 14092 bytes/sec = 845520 bytes**

Calculate basic figures:

*CDXL less audio:*

**9216000 - 845520 = 8370480 bytes**

*Number of frames:*

**8370480 bytes / 12168 bytes/frame = 688 frames**

Finally, calculate:

*Audio sample size per frame:*

**845520 bytes / 688 frames = 1228 bytes / frame**

Now you can proceed as follows:

*Use XLTrim to cut your grab down to 688 frames:*

**XLTrim -m688 video.grab video.688**

*With a program like "Audition4", resample your audio to 14K:*

**Resample to 14092 bytes/sec**

**Save file as audio.14092**

Finally, merge the video and audio data into a single file:

**XLAudio -a1228 audio.14092 video.688 final.xl**

Now, see how it looks and sounds:

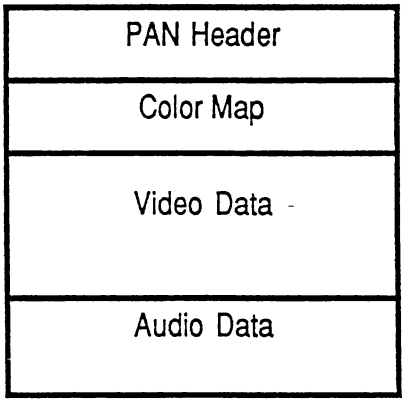
**XLPlay final.xl**

Keep in mind that there may be a few small timing errors when playing this on an Amiga. Normally you won't see these, but you may hear them. Unfortunately, the only way to be absolutely sure is to test with the emulator using your application program.

# Custom Programs

Within an application you will need code to play the CDXL sequences that you have created. You may also want to create special tools for manipulating CDXL sequences during the authoring process. To do this, you must understand the structure of a CDXL file and the philosophy behind it.

Standard CDXL files are made up of one or more frames. The general format of a frame is:




When a file contains more than one frame, the frames are simply concatenated, header and all. That is, one frame follows immediately after another. The only restriction is that frames must be aligned on even byte (word aligned) boundaries.

Various portions of a frame are optional. Actually, the only required section is the PAN Header, which contains information about the frame. The Color Map, Video Data, and Audio Data may or may not exist.

The PAN Structure is key to your access of CDXL files. It is defined in the file "pan.h" as:

```
struct PanFrame
{
    UBYTE Type;
```

This indicates what type of structure you are reading. Currently this field must be set to PAN_STANDARD.

```
    UBYTE Info;
```

This field indicates variations on the general Type. Various options are discussed below.

```
    ULONG Size;
```

This is the total size of the frame in bytes. It includes the entire PanFrame structure, color map, video, and audio. With this value, you can skip to the next frame.

```
ULONG Back;
```

This is the offset from the start of this frame back to the beginning of the previous frame. While CDXL cannot play backwards, this field is useful for tools.

```
ULONG Frame;
```

The Frame Sequence Number. First frame is number one (not zero). This number uniquely identifies the frame within a file. A zero value indicates that the number is undefined, and tools should print out a "??????" in the number field to show this.

```
UWORD XSize;
```

The width of the video in pixels.

```
UWORD YSize;
```

The height of the video in lines.

```
UBYTE Reserved;
```

Reserved for future. CURRENTLY, MUST BE ZERO!

```
UBYTE PixelSize;
```

The "depth" of a pixel. That is, the number of bits used to indicate a color (with or without a color map). A zero value indicates that no video data exists within the frame. Example values: 4 for 16 color, 6 for HAM, 24 for RGB24, etc.

```
UWORD ColorMapSize;
```

The total size of the color map in bytes. If you require more than 64K of color map, then you shouldn't be using CDXL (but you can use your own custom CDXL file, see below). A zero value indicates no color map. Typical values are 32 for 16 colors, 64 for 32 colors, 64 for half-bright (64 color) mode, 32 for HAM.

```
UWORD AudioSize;
```

Total size of the audio sample area in bytes. A zero indicates no audio is present. Total size must include all channels (when multiple audio channels are used).

```
UBYTE PadBytes[8];
```

This area is reserved for future use. For now it must always be set to zero.

```
};
```

The PAN Structure is optionally followed by the Color Map, Video data, and Audio data. Each of these areas must be even byte (16 bit word) aligned.

The Type field defined above is set to indicate the type of header structure being used. Over time, CDXL will evolve and this type will be critical for identifying what type of frame we are about to process. Right now the valid possibilities for this field are:

PAN_STANDARD – is used to indicate that the frame is in a standard format and that the above structure is used.

PAN_CUSTOM – is provided for you to create your own formats, but without causing problems for general tools. Custom formats allow you to lay your data however you need it. The only requirement is that the file start with the structure:

```
struct PanHead
{
   UBYTE Type;
```

Set to PAN_CUSTOM to identify this as nonstandard.

```
   UBYTE Info;
```

Reserved. Set to zero.

```
   ULONG Size;
```

This is the total size of the custom frame in bytes, including this structure. It is used to jump from the beginning of this frame to the beginning of the next frame.

```
};
```

PAN_SPECIAL – is reserved for experimental, and proposed formats.

All other values for the Type field are reserved for new frame structures. As CDXL evolves, new PAN types will be published.

In the PanFrame structure the info field describes various formats for the data contained within the frame.

PIV_MASK defines a bit field that tells you what video decoding should be used to display the image. Choices are:

PIV_STANDARD – for normal encoding (Amiga, RGB).

PIV_HAM – for hold and modify encoding

PIV_YUV – for luma-chroma encoding (eg. 844 bit YUV)

PIV_AVM – for CDTV advanced video mode and DCTV

Commodore will add new values to this as needed. Pixel value orientation is NOT specified here. It is defined in a separate bit field, PIF_MASK, that indicates the orientation format:

PIF_PLANES – data is interleaved on a bitplane basis, eg. in the normal Amiga bitplane style. Each plane contains one bit of the pixel value.

PIF_PIXELS – data is interleaved on per pixel basis, eg. the chunky style like that used for VGA data, 8 bits per pixel, 24 bits per pixel, whatever.

PIF_LINES – data is interleaved on a line basis. This can be thought of as each line being a single bitplane to itself.

Another field, PIA_MASK indicates audio options. It tells you whether the audio is mono or stereo.

A few size macros are provided to help you move through a file. These should be used as needed.

this document was
generously
contributed by

randell jesup

scanned by:
www.commodore.international